OSINT Python Tools

Summary:

Earlier this semester, I bought the book OSINT Techniques by Michael Bazzell because I wanted to start learning real open-source intelligence skills and eventually become an all-source intelligence analyst. The book inspired me to begin building my own simple OSINT tools so I could understand how information is collected, analyzed, and used in real investigations.

Since none of my group members participated in the project, I decided to complete the entire assignment on my own. Instead of doing random beginner Python exercises, I chose to create five small OSINT-focused Python tools. I felt this would not only meet the requirements of the assignment but also help me develop practical skills and start building a portfolio that connects directly to my future intelligence career. Each tool reflects something I learned from the book and turns it into a hands-on example I can continue improving over time.

With each project description, I included a potential use case to show the real-world value of the tool. This helps explain not just what the tool does, but why I chose to create it and how it could actually be applied in an OSINT or intelligence setting. By connecting each tool to a practical scenario, I wanted to demonstrate that these projects were intentionally designed to support the skills I need for my future career as an all-source analyst.

1. IP Address Geolocation Lookup

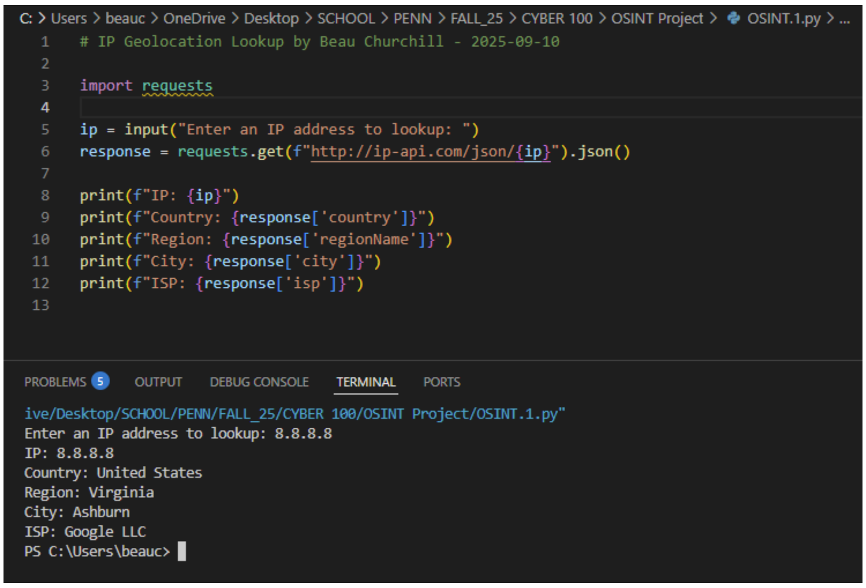
Description:
This tool takes an IP address and finds information about where it is located in the world. It can show the country, city, region, and the Internet Service Provider (ISP) that owns the IP. This is useful in intelligence because it helps analysts understand where network traffic is coming from, track suspicious activity, or investigate cyberattacks.

Use Case:
For example, if a company receives a series of phishing emails, an analyst can look up the IP addresses of the senders to see which countries the emails originated from. This can help identify patterns or potential threat actors.

Code:

```
# IP Geolocation Lookup by Beau Churchill - 2025-09-10

import requests

ip = input("Enter an IP address to lookup: ")
response = requests.get(f"http://ip-api.com/json/{ip}").json()

print(f"IP: {ip}")
print(f"Country: {response['country']}")
print(f"Region: {response['regionName']}")
print(f"City: {response['city']}")
print(f"ISP: {response['isp']}")
```

Screenshot:


C: > Users > beauc > OneDrive > Desktop > SCHOOL > PENN > FALL_25 > CYBER 100 > OSINT Project > OSINT.1.py > ...
PROBLEMS 5
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

```
# IP Geolocation Lookup by Beau Churchill - 2025-09-10

import requests

ip = input("Enter an IP address to lookup: ")
response = requests.get(f"http://ip-api.com/json/{ip}").json()

print(f"IP: {ip}")
print(f"Country: {response['country']}")
print(f"Region: {response['regionName']}")
print(f"City: {response['city']}")
print(f"ISP: {response['isp']}")
```

```
ive/Desktop/SCHOOL/PENN/FALL_25/CYBER 100/OSINT Project/OSINT.1.py"
Enter an IP address to lookup: 8.8.8.8
IP: 8.8.8.8
Country: United States
Region: Virginia
City: Ashburn
ISP: Google LLC
PS C:\Users\beauc>
```

2. Domain WHOIS Lookup

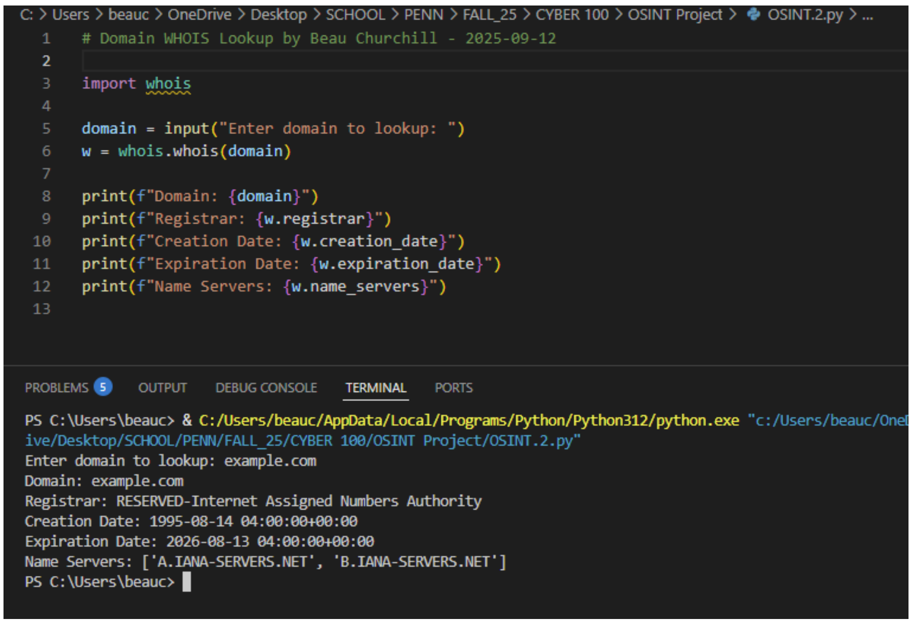
Description:
A WHOIS lookup retrieves registration information about a website or domain. This includes the registrar, creation date, expiration date, and sometimes contact information. Intelligence analysts use this information to understand who owns a domain and when it was created, which can help detect fake or suspicious websites.

Use Case:
If an unknown website is spreading malware or phishing emails, an analyst can run a WHOIS lookup to see when it was created and who registered it. New domains with hidden or unusual registration information may indicate malicious intent.

Code:

```
# Domain WHOIS Lookup by Beau Churchill - 2025-09-12

import whois

domain = input("Enter domain to lookup: ")
w = whois.whois(domain)

print(f"Domain: {domain}")
print(f"Registrar: {w.registrar}")
print(f"Creation Date: {w.creation_date}")
print(f"Expiration Date: {w.expiration_date}")
print(f"Name Servers: {w.name_servers}")
```

Screenshot:


C: > Users > beauc > OneDrive > Desktop > SCHOOL > PENN > FALL_25 > CYBER 100 > OSINT Project > OSINT.2.py > ...
PROBLEMS 5
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

```
# Domain WHOIS Lookup by Beau Churchill - 2025-09-12

import whois

domain = input("Enter domain to lookup: ")
w = whois.whois(domain)

print(f"Domain: {domain}")
print(f"Registrar: {w.registrar}")
print(f"Creation Date: {w.creation_date}")
print(f"Expiration Date: {w.expiration_date}")
print(f"Name Servers: {w.name_servers}")
```

```
PS C:\Users\beauc> & C:/Users/beauc/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/beauc/OneD
ive/Desktop/SCHOOL/PENN/FALL_25/CYBER 100/OSINT Project/OSINT.2.py"
Enter domain to lookup: example.com
Domain: example.com
Registrar: RESERVED-Internet Assigned Numbers Authority
Creation Date: 1995-08-14 04:00:00+00:00
Expiration Date: 2026-08-13 04:00:00+00:00
Name Servers: ['A.IANA-SERVERS.NET', 'B.IANA-SERVERS.NET']
PS C:\Users\beauc> █
```

3. Social Media Profile Scraper

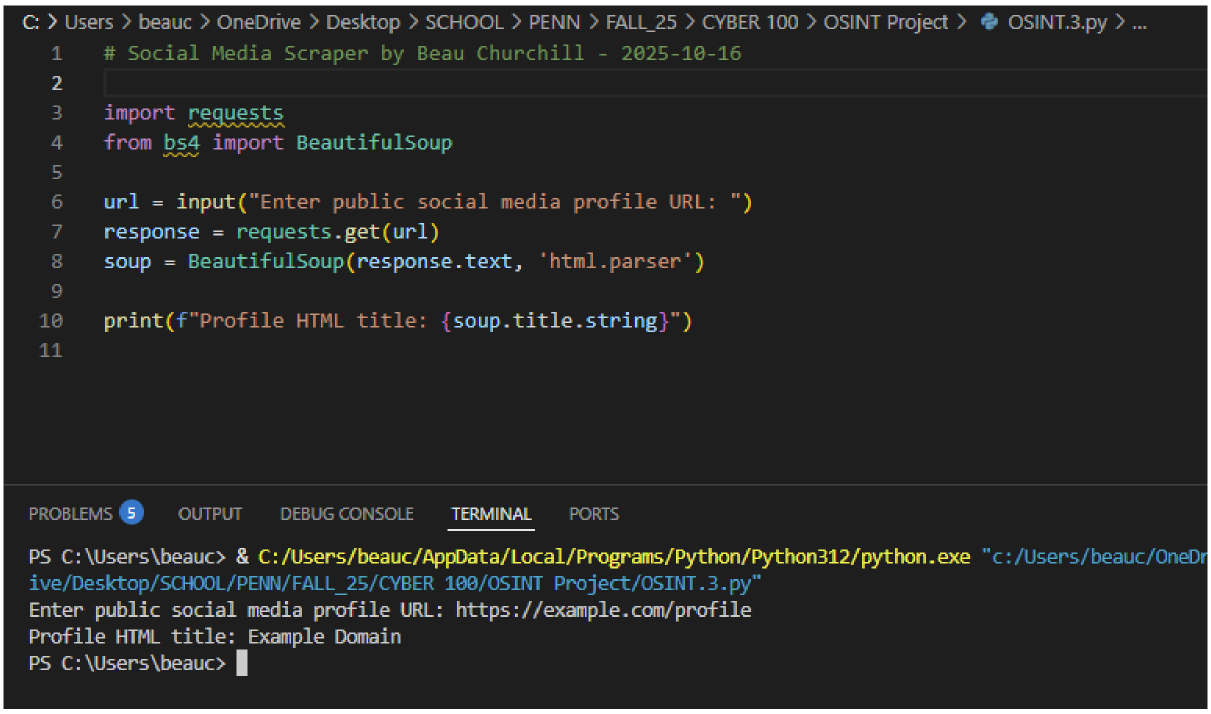
Description:
This tool reads publicly available information from a social media profile. It can show profile titles, page content, or basic information about the account. Analysts can use this to gather OSINT (open-source intelligence) without accessing private data.

Use Case:
Suppose an analyst is investigating a potential insider threat. They can use this tool to examine the public social media activity of the person, looking for posts or patterns that might indicate security risks or behavioral clues.

Code:

```
# Social Media Scraper by Beau Churchill - 2025-10-16

import requests
from bs4 import BeautifulSoup

url = input("Enter public social media profile URL: ")
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

print(f"Profile HTML title: {soup.title.string}")
```

Screenshot:


C: > Users > beauc > OneDrive > Desktop > SCHOOL > PENN > FALL_25 > CYBER 100 > OSINT Project > OSINT.3.py > ...
PROBLEMS 5
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

```
# Social Media Scraper by Beau Churchill - 2025-10-16

import requests
from bs4 import BeautifulSoup

url = input("Enter public social media profile URL: ")
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

print(f"Profile HTML title: {soup.title.string}")
```

```
PS C:\Users\beauc> & C:/Users/beauc/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/beauc/OneDr
ive/Desktop/SCHOOL/PENN/FALL_25/CYBER 100/OSINT Project/OSINT.3.py"
Enter public social media profile URL: https://example.com/profile
Profile HTML title: Example Domain
PS C:\Users\beauc>
```

4. Email Verification Tool

Description:

This program checks if an email address has a valid format and belongs to a real domain. It is helpful for intelligence work because it can verify the reliability of email contacts or detect potentially fake emails that are used in scams and phishing attacks.

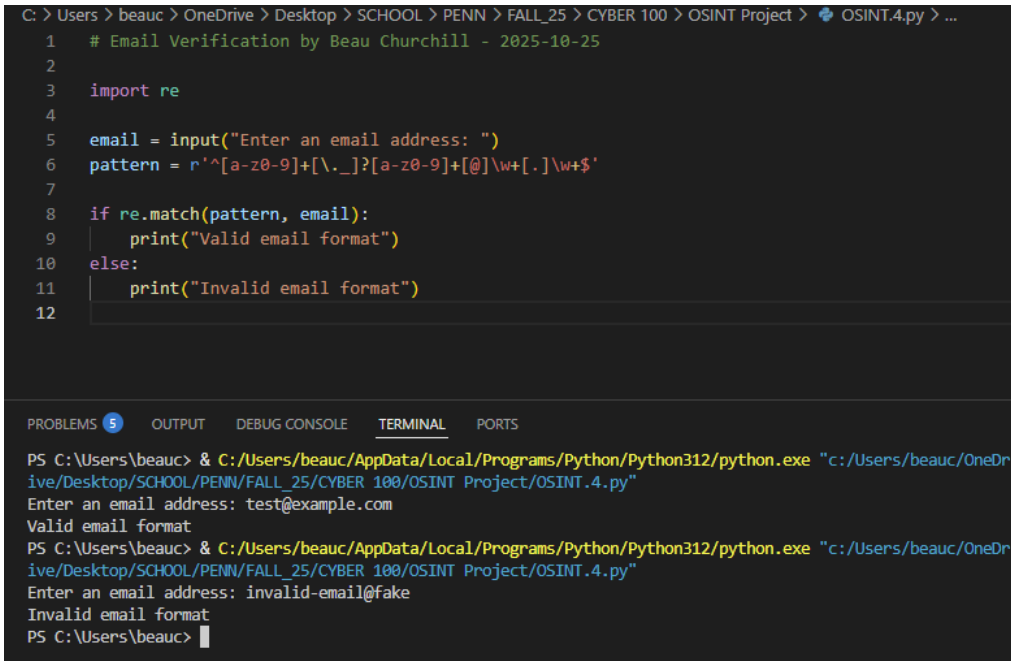
Use Case:

An analyst might receive suspicious emails claiming to be from a trusted source. Using this tool, they can check if the sender's email address looks real and if it comes from a legitimate domain before taking further action.

Code:

```
# Email Verification by Beau Churchill - 2025-10-25

import re

email = input("Enter an email address: ")
pattern = r'^[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w+$'

if re.match(pattern, email):
    print("Valid email format")
else:
    print("Invalid email format")
```

Screenshot:



C: > Users > beauc > OneDrive > Desktop > SCHOOL > PENN > FALL_25 > CYBER 100 > OSINT Project > OSINT.4.py > ...

```
# Email Verification by Beau Churchill - 2025-10-25

import re

email = input("Enter an email address: ")
pattern = r'^[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w+$'

if re.match(pattern, email):
    print("Valid email format")
else:
    print("Invalid email format")

```

```
PS C:\Users\beauc> & C:/Users/beauc/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/beauc/OneDr
ive/Desktop/SCHOOL/PENN/FALL_25/CYBER 100/OSINT Project/OSINT.4.py"
Enter an email address: test@example.com
Valid email format
PS C:\Users\beauc> & C:/Users/beauc/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/beauc/OneDr
ive/Desktop/SCHOOL/PENN/FALL_25/CYBER 100/OSINT Project/OSINT.4.py"
Enter an email address: invalid-email@fake
Invalid email format
PS C:\Users\beauc> █
```

5. Brave Search Automation

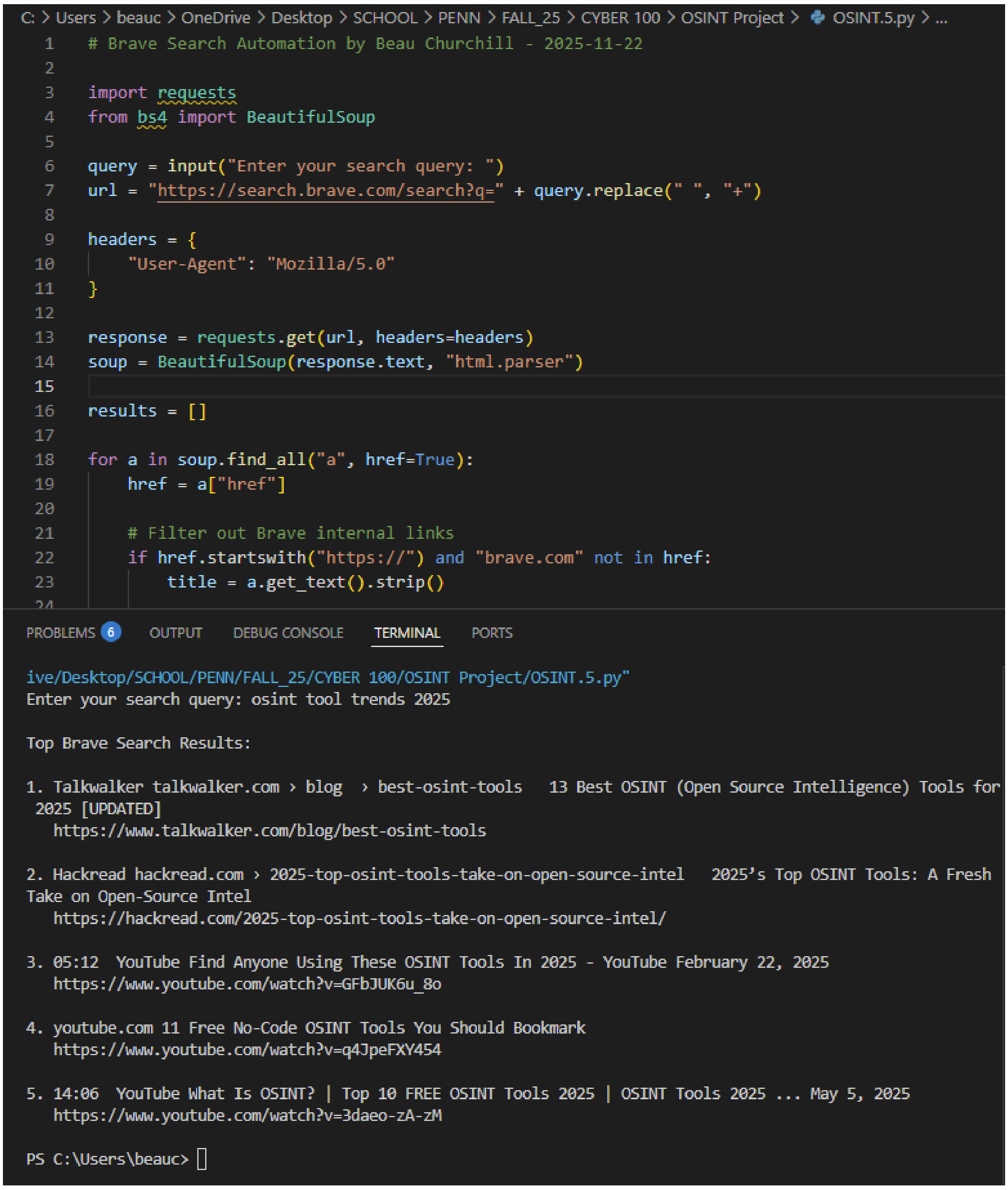
Description:
This tool automates Brave searches (as Google doesn't allow scraping without a specific API) and collects the top results for a given query. Analysts use it to quickly gather information from open sources about a person, organization, or event. It helps find relevant websites, news articles, and public data efficiently.

Use Case:
If an intelligence officer needs to gather background information on a company suspected of illegal activity, they can run a search query about the company and automatically collect the top results. This saves time and ensures no important sources are missed.

Code:

```
# Brave Search Automation by Beau Churchill - 2025-11-22

import requests
from bs4 import BeautifulSoup

query = input("Enter your search query: ")
url = "https://search.brave.com/search?q=" + query.replace(" ", "+")

headers = {
    "User-Agent": "Mozilla/5.0"
}

response = requests.get(url, headers=headers)
soup = BeautifulSoup(response.text, "html.parser")

results = []

for a in soup.find_all("a", href=True):
    href = a["href"]

    # Filter out Brave internal links
    if href.startswith("https://") and "brave.com" not in href:
        title = a.get_text().strip()

        # Skip empty titles and images
        if title and len(title) > 3:
            results.append((title, href))

print("\nTop Brave Search Results:\n")

if not results:
    print("No results found. (Brave HTML may have changed again.)")
else:
    for i, (title, link) in enumerate(results[:5], start=1):
        print(f"{i}. {title}\n   {link}\n")
```

Screenshot:



C: > Users > beauc > OneDrive > Desktop > SCHOOL > PENN > FALL_25 > CYBER 100 > OSINT Project > OSINT.5.py > ...

```python
# Brave Search Automation by Beau Churchill - 2025-11-22

import requests
from bs4 import BeautifulSoup

query = input("Enter your search query: ")
url = "https://search.brave.com/search?q=" + query.replace(" ", "+")

headers = {
    "User-Agent": "Mozilla/5.0"
}

response = requests.get(url, headers=headers)
soup = BeautifulSoup(response.text, "html.parser")

results = []

for a in soup.find_all("a", href=True):
    href = a["href"]

    # Filter out Brave internal links
    if href.startswith("https://") and "brave.com" not in href:
        title = a.get_text().strip()
```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
ive/Desktop/SCHOOL/PENN/FALL_25/CYBER 100/OSINT Project/OSINT.5.py"
Enter your search query: osint tool trends 2025

Top Brave Search Results:

1. Talkwalker talkwalker.com › blog  › best-osint-tools   13 Best OSINT (Open Source Intelligence) Tools for
 2025 [UPDATED]
   https://www.talkwalker.com/blog/best-osint-tools

2. Hackread hackread.com › 2025-top-osint-tools-take-on-open-source-intel   2025’s Top OSINT Tools: A Fresh
Take on Open-Source Intel
   https://hackread.com/2025-top-osint-tools-take-on-open-source-intel/

3. 05:12  YouTube Find Anyone Using These OSINT Tools In 2025 - YouTube February 22, 2025
   https://www.youtube.com/watch?v=GFbJUK6u_8o

4. youtube.com 11 Free No-Code OSINT Tools You Should Bookmark
   https://www.youtube.com/watch?v=q4JpeFXY454

5. 14:06  YouTube What Is OSINT? | Top 10 FREE OSINT Tools 2025 | OSINT Tools 2025 ... May 5, 2025
   https://www.youtube.com/watch?v=3daeo-zA-zM

PS C:\Users\beauc> 
```